TDTS06

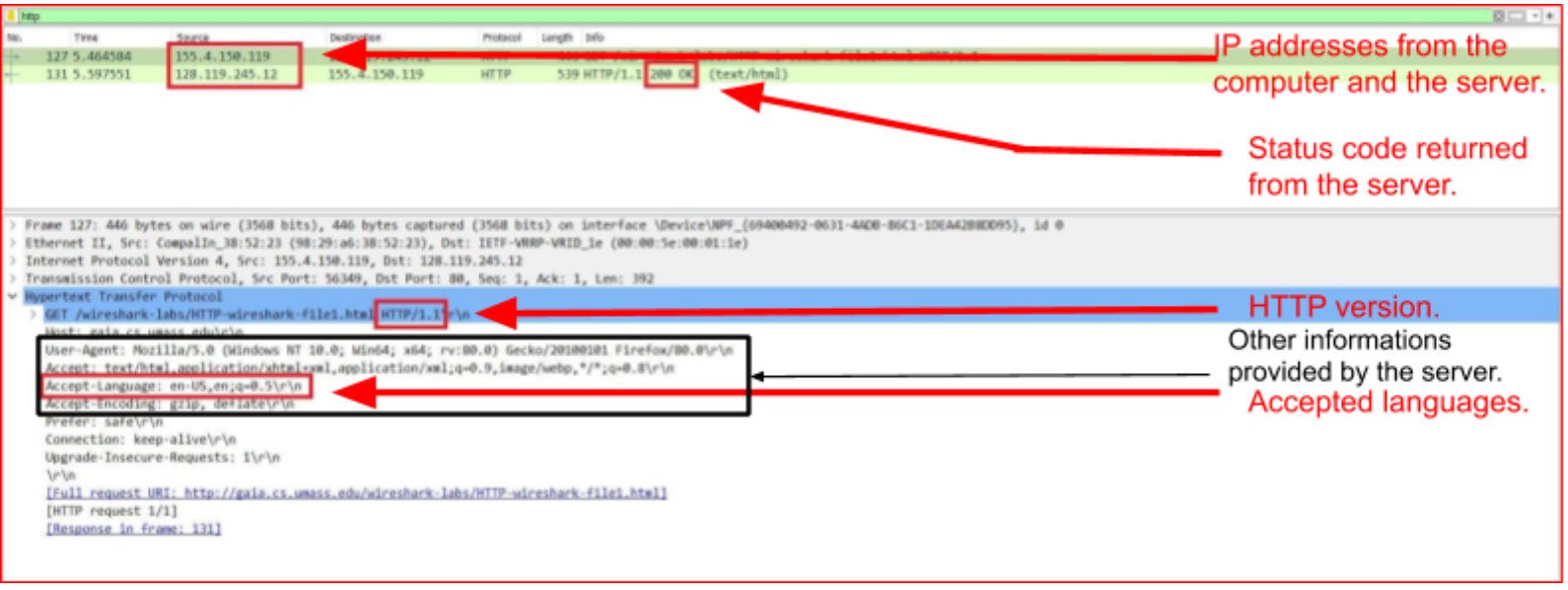
Lab 1 report

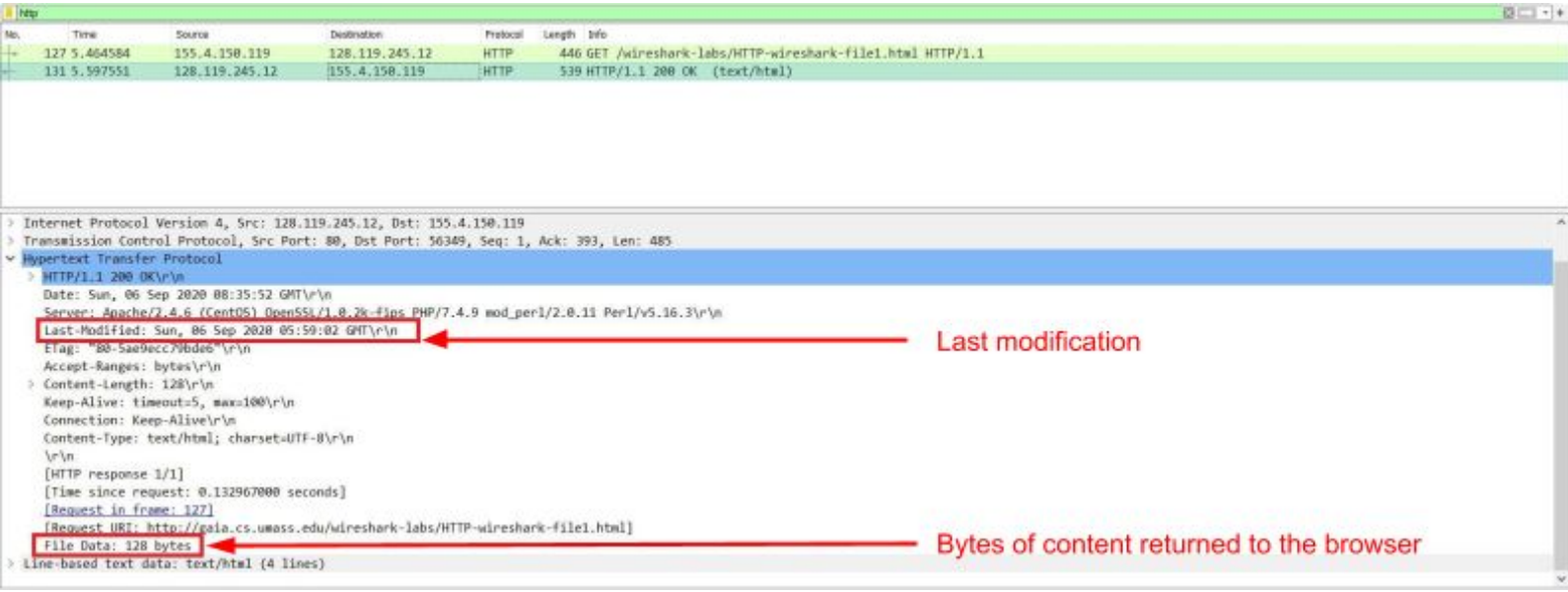
Mayeul Gasc (mayga330@student.liu.se)
Pierre Marsaa (piema262@student.liu.se)

Task A

1) We can read in the GET method in the first packet '*Request Version: HTTP/1.1*' we deduce that the HTTP version of the client is 1.1. The following package gives us the version used by the server '*Response Version: HTTP/1.1*'', also version 1.1.
2) The first GET request from the client gives us several information about the client's browser. First, the languages accepted by the browser '*Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.5,en;q=0.3*' which means that the desired language is France french, or failing that Basic french, US english or english (in this order of priority). The packet also gives information about the accepted compression methods, the browser used, the operating system and accepted type of data (html, xml, etc.).
3) The client being the one using the GET method, we deduce from the first packet that the address of the client (the source) is 155.4.151.182 and that of the server (the destination) is 128.119.245.12.
4) The returned status code is 200, which means that the request was successful.
5) The last modification on the page is indicated as having taken place approximately 7 minutes before the request.
6) The packet contains the number of content bytes returned by the server : 128 bytes.
7) No, there isn't.

In this first part, we saw how a browser obtains a web page with the HTTP protocol. A GET request is sent with all the necessary information to the server to provide an appropriate response. We also notice that if we try to load the page again, the response from the server changes and indicates that there has been no modification and that the browser can try to load the cache (code 304). This is probably done to limit the consumption of bandwidth.

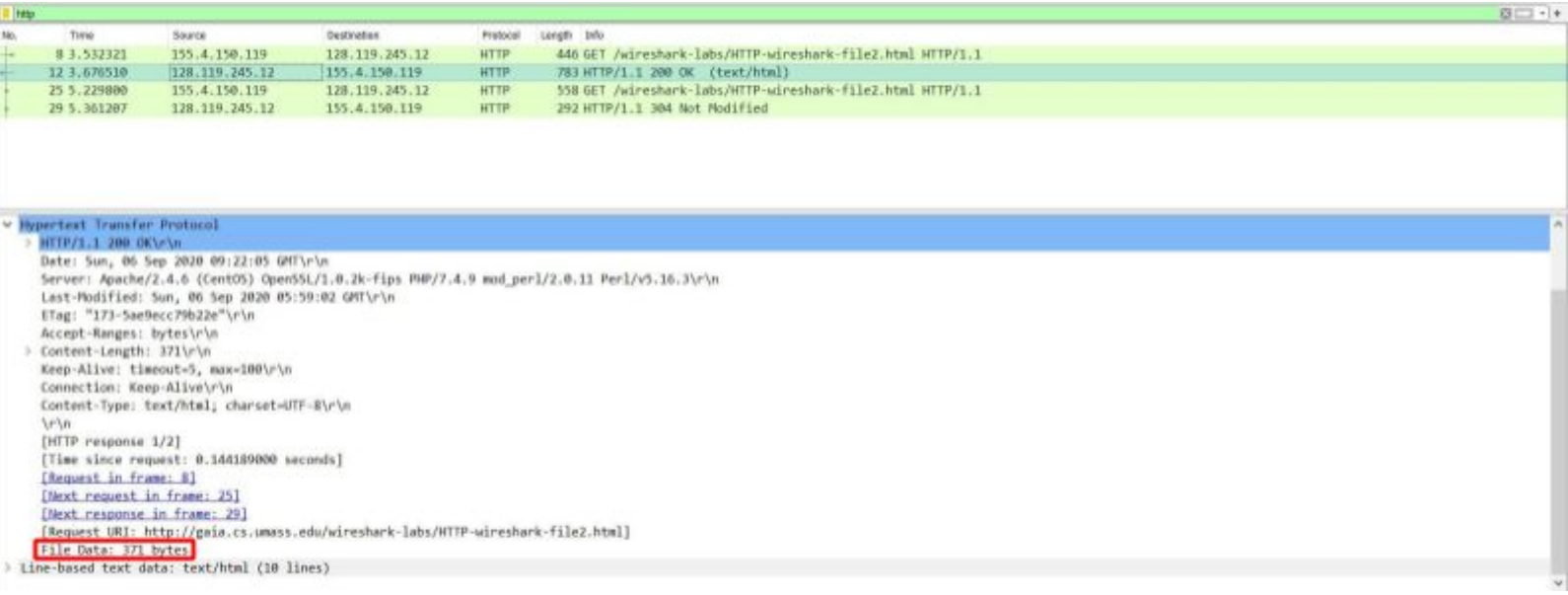


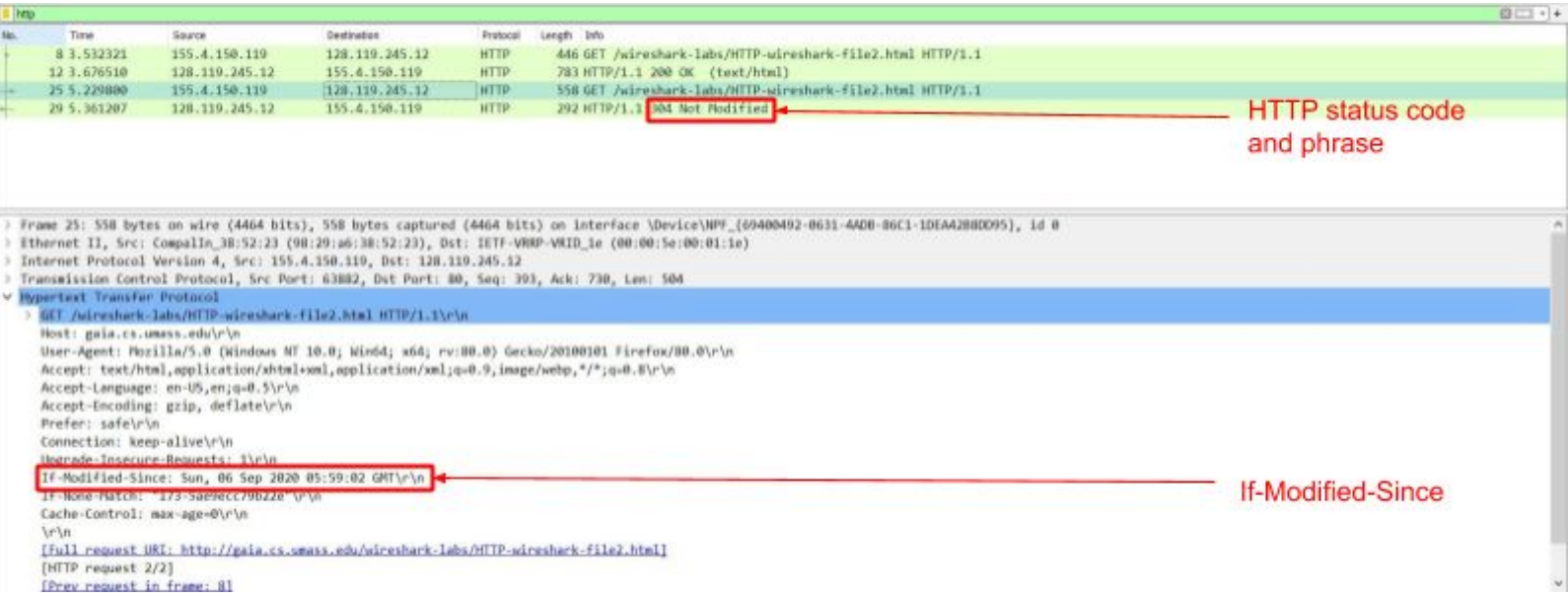


Task B

8) No, there is no if-modified-since line.
9) Yes, the server clearly sent the entire page, because we see a 371 byte "data" field which corresponds to an html page.
10) The second request has the if-modified-since field with the date of the first packet.
11) While the first request gets the same response as in part A, the second gets a smaller response than the first time with the code 304 "not modified". We understand here that the browser asked the server if the page has been modified and that the latter answered no. The browser therefore loaded the cached page.

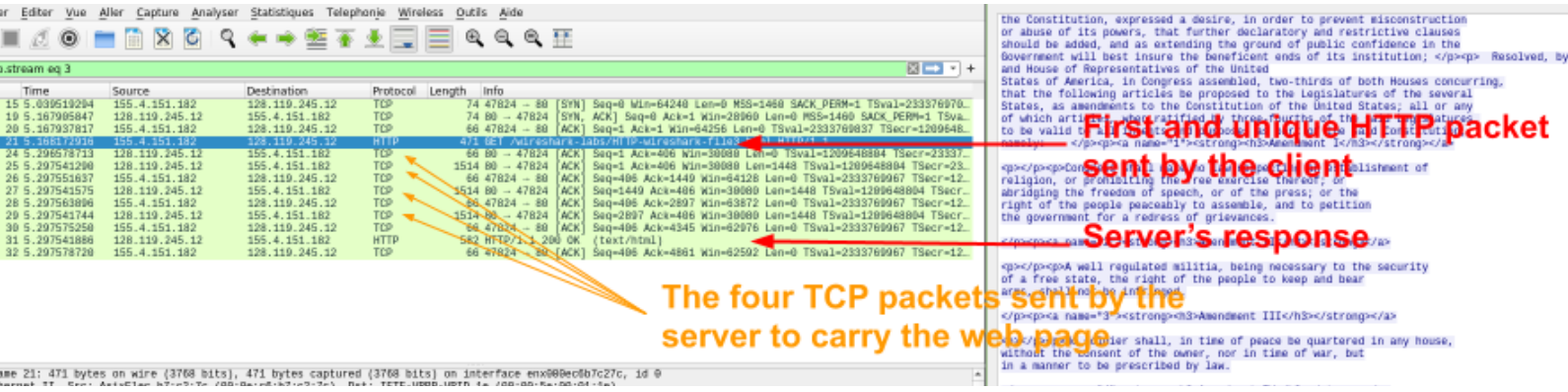
We observe in this part that the HTTP protocol is designed so as not to transfer unnecessary data. Indeed, the browser when it requests a page from the server, it will keep it in cache and in case the user requests the same page again, it specifies in the request that it already has a cached page. If no changes have taken place, the page is not resent and code 302 sent. The browser then displays the cached page.





Task C

12) The browser sends only one GET request, the number of the packet is 543.
13) The packet associated with the response is 559 with the code 200 (request OK).
14) It took four TCP packets to transfer the web page. (only packet containing the segmented data)
15) Since the HTTP and TCP protocols are not at the same layer, the HTTP is encapsulated in the TCP, so there is no communication between the two.

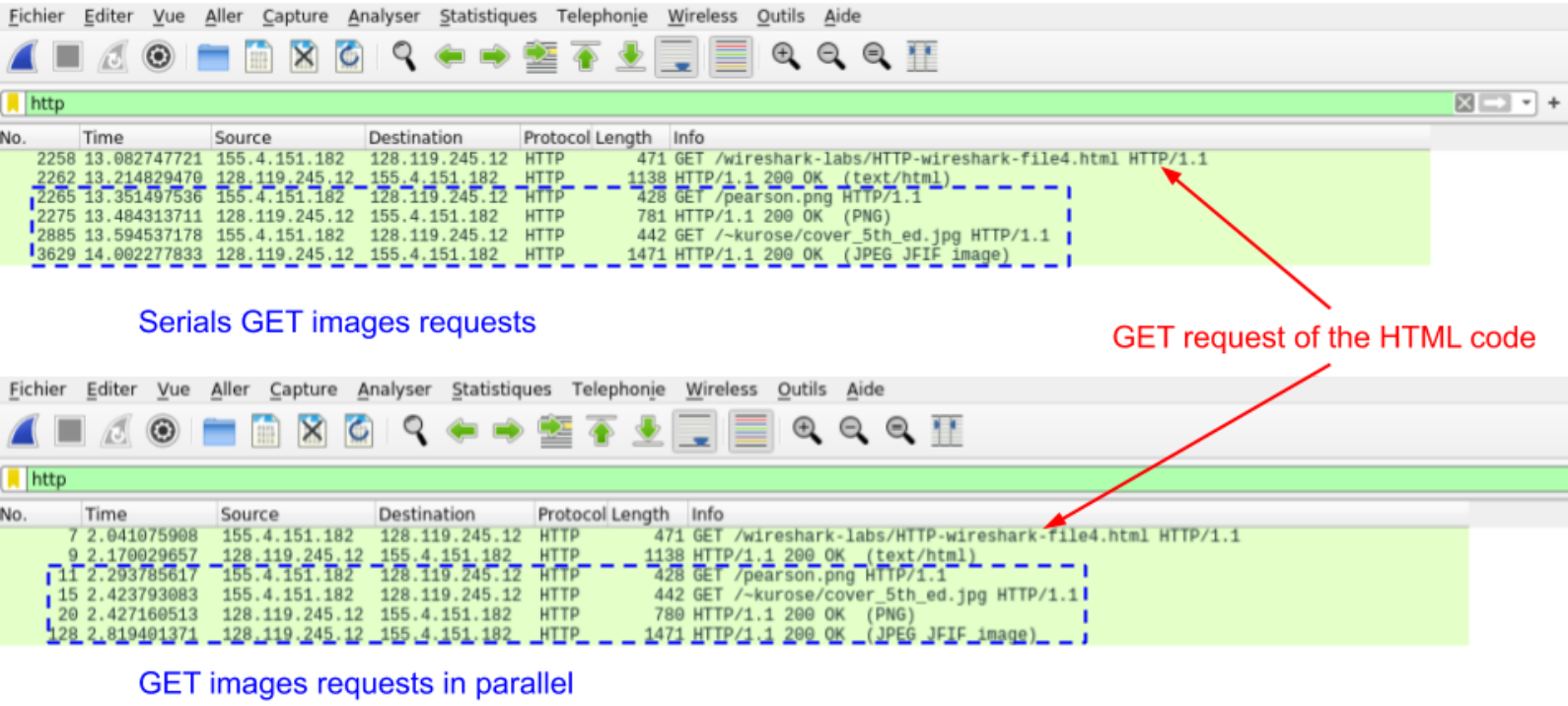


In this part, we've seen that some HTTP responses are too big to fit in one packet. That's why the responses can be divided into a certain number of TCP packets. The number of TCP packets depend on the size of the HTTP response (1460 bytes per packet). Both protocols aren't on the same layer and therefore do not communicate with each other. It is only the TCP protocol that should be concerned with dividing the message into packets.

Task D

16) Three requests were sent by the browser, one for the HTML code of the page and two others for the two images. All requests are sent to the site gaia.cs.umass.edu (128.119.245.12).
17) The two requests were sent in parallel, we notice on the capture that the GET requests are grouped, as well as the responses to these requests.

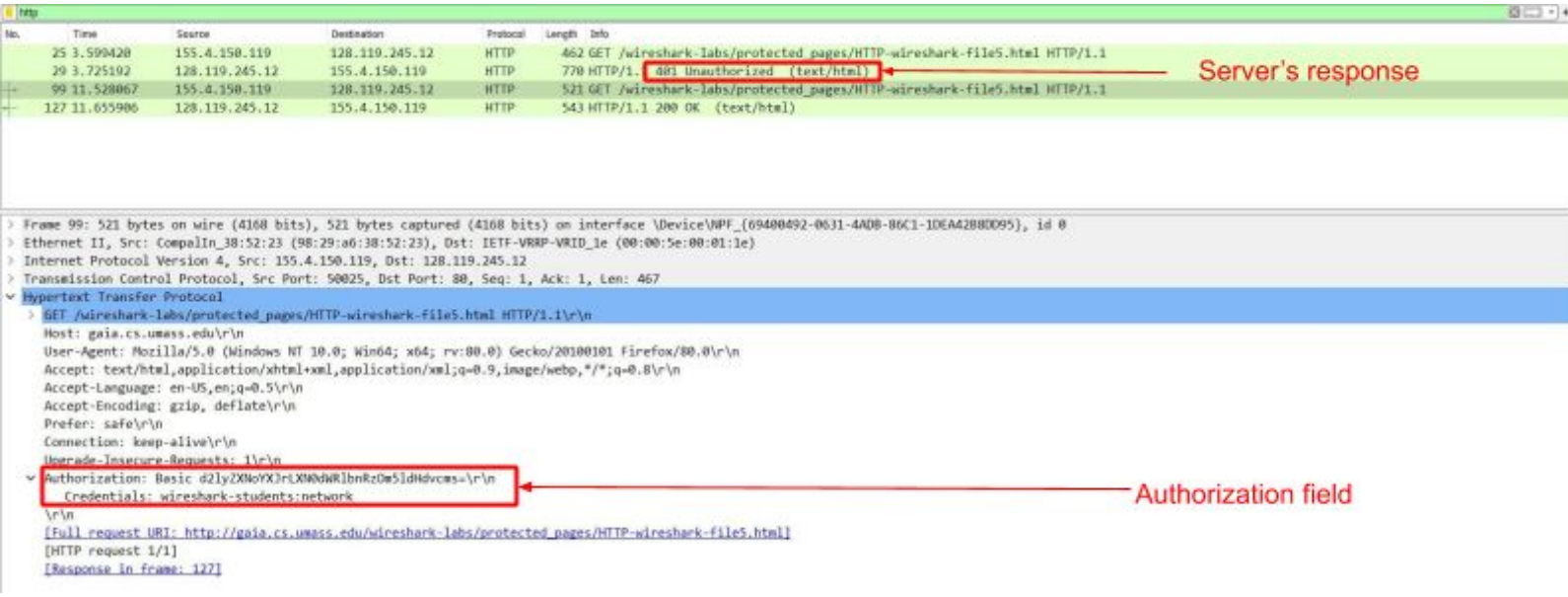
With this task, we saw that each image needed one request for itself and that the request can be done in parallel (twice get, twice reponses) or serially (one get, one response, one get, one response). During our various tests, we observed both methods.



Task E

18) The server's first response was an access denied with the code 401.
19) In the second request, a new line was there: 'Authorization' (with the connection information entered)

Here we have an example of authentication via HTTP protocol. The client first asks to access a web page and the server responds the first time indicating access denied. The browser then invites the user to enter the identifiers that it sends to the server in a new request. If the



credentials are correct, the response from the server contains the desired web page. We notice that the identifiers are sent in clear and that there is a security problem.

HTTP Persistent connection

20) According to our research on the internet, these fields indicate whether the HTTP connection should be maintained or not. With version 1.1 of the HTTP protocol, the connection is not systematically closed after the end of the request so as not to have to open a new one if browsing continues on the same website.